Amendments to the Claims:

This listing of claims will replace all prior versions, and listings, of claims in the application.

1. – 46. (Canceled).

47. (New) A method, comprising:

modifying methods of a class file to include additional byte code instructions at said methods' respective entry and/or exit points;

modifying said class file to include a registration method information structure;

registering said class file with a dispatcher, said registering of said class file occurring after said additional byte code instructions and said registration method information structure have been added to said class file, said registering including passing said class file's name, said methods' names and said methods' argument types to said dispatcher from said registration method information structure;

said dispatcher performing the following in response to said registering of said class file:

identifying a user defined plug-in pattern for said class file that lists one or more plug-ins for each one of said class-file's methods;

updating a dictionary maintained by said dispatcher with said plug-in pattern;

passing an identifier to said class file that uniquely identifies said class file;

executing said class file's methods in a run-time environment, said methods invoking said dispatcher upon reaching said entry and/or exit points, each invocation passing to said dispatcher: a) said identifier, b) an identity of the invoking method, and, c) the invoking method's arguments;

said dispatcher, for each invocation, using a), b) and c) above to look-up an appropriate plug-in from said dictionary and passing an identity of said appropriate

plug-in to said invoking method so that said invoking method can receive appropriate handler treatment from said appropriate plug-in.

48. (New) The method of claim 47 wherein said updating of said dictionary includes:

adding to said dictionary a first entry that correlates a first plug-in with an entry point of a first method of said class file;

adding to said dictionary a second entry that correlates a second plug-in with an exit point of a second method of said class file.

49. (New) The method of claim 48 wherein said identifier is determined by incrementing a numeric value that was passed to a second class file that registered with said dispatcher immediately before said class file registered with said dispatcher.

50. (New) The method of claim 49 wherein said first and second plug-ins are the same plug-in.

51. (New) The method of claim 50 wherein said plug-in contains a handler having a counter to count method entries and method exits.

52. (New) A computing system comprising one or more processors coupled to memory, said memory storing program code, said one or more processors to process said program code to perform a method, said method comprising:

modifying methods of a class file to include additional byte code instructions at said methods' respective entry and/or exit points;

modifying said class file to include a registration method information structure;

registering said class file with a dispatcher, said registering of said class file occurring after said additional byte code instructions and said registration method information structure have been added to said class file, said registering including

passing said class file's name, said methods' names and said methods' argument types to said dispatcher from said registration method information structure;

said dispatcher performing the following in response to said registering of said class file:

identifying a user defined plug-in pattern for said class file that lists one or more plug-ins for each one of said class-file's methods;

updating a dictionary maintained by said dispatcher with said plug-in pattern;

passing an identifier to said class file that uniquely identifies said class file;

executing said class file's methods in a run-time environment, said methods invoking said dispatcher upon reaching said entry and/or exit points, each invocation passing to said dispatcher: a) said identifier, b) an identity of the invoking method, and, c) the invoking method's arguments;

said dispatcher, for each invocation, using a), b) and c) above to look-up an appropriate plug-in from said dictionary and passing an identity of said appropriate plug-in to said invoking method so that said invoking method can receive appropriate handler treatment from said appropriate plug-in.

53. (New) The computing system of claim 52 wherein said updating of said dictionary includes:

adding to said dictionary a first entry that correlates a first plug-in with an entry point of a first method of said class file;

adding to said dictionary a second entry that correlates a second plug-in with an exit point of a second method of said class file.

54. (New) The computing system of claim 53 wherein said identifier is determined by incrementing a numeric value that was passed to a second class file that registered with said dispatcher immediately before said class file registered with said dispatcher.

55. (New) The computing system of claim 54 wherein said first and second plug-ins are the same plug-in.

56. (New) The computing system of claim 55 wherein said plug-in contains a handler having a counter to count method entries and method exits.

57. (New) A machine readable storage medium containing program code to be executed by one or more processors of a computing system, said one or more processors comprising:

modifying methods of a class file to include additional byte code instructions at said methods' respective entry and/or exit points;

modifying said class file to include a registration method information structure;

registering said class file with a dispatcher, said registering of said class file occurring after said additional byte code instructions and said registration method information structure have been added to said class file, said registering including passing said class file's name, said methods' names and said methods' argument types to said dispatcher from said registration method information structure;

said dispatcher performing the following in response to said registering of said class file:

identifying a user defined plug-in pattern for said class file that lists one or more plug-ins for each one of said class-file's methods;

updating a dictionary maintained by said dispatcher with said plug-in pattern;

passing an identifier to said class file that uniquely identifies said class file;

executing said class file's methods in a run-time environment, said methods invoking said dispatcher upon reaching said entry and/or exit points, each invocation passing to said dispatcher: a) said identifier, b) an identity of the invoking method, and, c) the invoking method's arguments;

said dispatcher, for each invocation, using a), b) and c) above to look-up an appropriate plug-in from said dictionary and passing an identity of said appropriate

plug-in to said invoking method so that said invoking method can receive appropriate handler treatment from said appropriate plug-in.

58. (New) The machine readable storage medium of claim 57 wherein said updating of said dictionary includes:

adding to said dictionary a first entry that correlates a first plug-in with an entry point of a first method of said class file;

adding to said dictionary a second entry that correlates a second plug-in with an exit point of a second method of said class file.

59. (New) The machine readable storage medium of claim 48 wherein said identifier is determined by incrementing a numeric value that was passed to a second class file that registered with said dispatcher immediately before said class file registered with said dispatcher.

60. (New) The machine readable storage medium of claim 59 wherein said first and second plug-ins are the same plug-in.

61. (New) The machine readable storage medium of claim 60 wherein said plug-in contains a handler having a counter to count method entries and method exits.